Noname manuscript No.
(will be inserted by the editor)

Knowledge Representation in Learning Classifier Systems: A Survey

Farzaneh Shoeleh · Mahshid Majd · Ali Hamzeh · Sattar Hashemi

Received: date / Accepted: date

Abstract Knowledge representation is a key component to the success of all rule based systems including learning classifier systems (LCSs). This component brings insight into how to partition the problem space what in turn seeks prominent role in generalization capacity of the system as a whole. Recently, knowledge representation component has received great deal of attention within data mining communities due to its impacts on rule based systems in terms of efficiency and efficacy. The current work is an attempt to find a comprehensive and yet elaborate view into the existing knowledge representation techniques in LCS domain in general and XCS in specific.

To achieve the objectives, knowledge representation techniques are grouped into different categories based on the classification approach in which they are incorporated. In each category, the underlying rule representation schema and the format of classifier condition to support the corresponding representation are presented. Furthermore, a precise explanation on the way that each technique partitions the problem space along with the extensive experimental results is provided. To have an elaborated view on the functionality of each technique, a comparative analysis of existing techniques on some conventional problems is provided. We expect this survey to be of interest to the LCS researchers and practitioners since it provides a guideline for choosing a proper knowledge representation technique for a given problem and also opens up new streams of research on this topic.

Keywords Learning Classifier Systems · XCS · Knowledge Representation · Rule Based Systems

F. Shoeleh
Computer Science and Engineering Dept. Shiraz University, Shiraz, Iran.
E-mail: shoeleh@cse.shirazu.ac.ir

M. Majd
E-mail: majd@cse.shirazu.ac.ir

A. Hamzeh
E-mail: ali@cse.shirazu.ac.ir

S. Hashemi
E-mail: s_hashemi@shirazu.ac.ir

1 Introduction

The first framework of Learning Classifier System (LCS) labeled ”cognitive system” was introduced more than 30 years ago by John H. Holland (Holand, 1976). LCSs were originally inspired by the general principles of Darwinian evolution and cognitive learning. The LCS framework was reformed to use reinforcement learning techniques such as Q-learning (Sutton and Barto, 1998) in order to ensure appropriate reward estimation and propagation. LCS is also known as rule-based evolutionary online learning system. It is a heuristic method in which a population of production systems are consisted and adapted by using genetic algorithm and reinforcement learning techniques. Each production system can cover small region of environment and represent some portions of the overall solution. Therefore, a LCS system is able to solve a problem by using the best evolved production systems in its population.

According to the solution encoding methodology, LCS has been addressed from two different points of view: the Pittsburgh Classifier System (Smith, 1980), usually referring to the models which have been inspired by the work of Smith and De Jong at the University of Pittsburg (Smith, 1980; De Jong, 1988; Smith, 1983), and Michigan Classifiers Systems that are usually the models which have been inspired by Holland's work at the University of Michigan (Holand, 1976; Holand, 1995). In Pittsburg approach, each population member is a production system and GA selects the best one as a complete solution of a given problem. In Michigan approach, the population members are individual rules, and the whole population forms the solution of the given problem. A rewarding mechanism is needed to reward and penalize bad rules.

The research in learning classifier systems sprang up in the 1980s. Early classifier systems were known as modeling tools. But they lost this initial characterization with rise of the abilities in the area of machine learning, especially in classification problems and the interest in reinforcement learning and autonomous agents. The year 1995 marked as a milestone in LCS researches due to Wilson's flavor of Holland's recipe. Wilson revised the main structure of Holland's LCS by simplifying it and changing its learning mechanism to use reinforcement learning techniques introduced in (Sutton and Barto, 1998). Nowadays the most popular Michigan system is an evolution of zeroth level classifier system, ZCS (Wilson, 1994), called accuracy based learning classifier system, XCS (Wilson, 1995), which can be considered as a milestone in classifier system research. XCS was the first LCS wherein the fitness of each rule is defined based on its accuracy of the payoff prediction instead of the receiving reward itself. Also, it has another main feature which uses the action set to define the environment niches. In Lanzi's view, the effectiveness of XCS as a machine learning paradigm is that *"XCS was the first classifier system to be both general enough to allow applications to several domains and simple enough to allow duplication of the presented results"* (Lanzi, 2008).

From 1995 to now, the applicability of XCS has been extended to a wide range of applications including computational economics (Schulenburg and Ross, 1999; Wong and Schulenburg, 2007), classification and data mining (Bull et al., 2008), autonomous robotics (Patel and Dorigo, 1994; Studley, 2005; Studley and Bull, 2005), power distribution network (Vargas et al., 2004), traffic light control (Bull et al., 2004), function approximation tasks (Wilson, 2002) and many more (Bull, 2004; Lanzi et al., 2000; Bull et al., 2008). So, the current LCS researches are very diverse and pervasive and done better than ever.

There are many valuable researches to show the effectiveness and the power of LCS in many domains. Some of these researches provided a description of the overall advances of the LCS field as a survey study. For example, in (Lanzi, 2008), the author tried to answer this question *"What has happened to learning classifier systems in the last decade?"* and examined the current state of learning classifier systems research. Indeed, the aim of this study is to emphasize recent developments and the state-of-the-art Learning Classifier Systems. Bull et al. brought together the works on the use of LCS for data mining problems (Bull et al., 2008). LCSs have proved its efficiency at solving online and offline classification tasks. Bacardit et al. gave a summary of past, present and future LCS researches in (Bacardit et al., 2008). They tried to take a look back at the LCS realm and discuss which challenges and opportunities are laying ahead for successful system applications in various domains. As LCS can be seen as a learning system which is able to solve reinforcement learning problems, a study (Sigaud and Wilson, 2007) was done which focuses more on the sequential decision domains than on automatic classification. In (Urbanowicz and Moore, 2009), after introducing a description of basic LCS framework, a historical review of major advancements and a roadmap of algorithmic components are provided. It also emphasizes the differences between alternative LCS implementation and their problem domains. The main aim of such studies is to provide accessible and comprehensible principles and different backgrounds for researches interested in developing their own LCS to achieve a specific goal.

To the best of our knowledge, most of the existing surveys have tried to investigate the general behavior of LCS in terms of its common issues, basic components and major applications. These studies can be considered as appropriate resources for who are interested in LCS research field providing that they give a general perspective of the current research opportunities and challenges in LCSs area. But they might not be efficient enough to show the progress of LCSs regarding to the improvement of their main components to handle new issues and challenges. In other words, according to the large number of studies dedicated to each of the main components of LCSs, surveys individually focusing on each one would be of great use provided that they show how such component can be modified to improve the performance of a LCS like XCS in solving a given problem. Consequently, this survey attempts to provide a detailed description of LCS concentrating on one of its major components known as knowledge representation. . Recently, knowledge representation component has received great deal of attention within data mining communities due to its impacts on rule based systems in terms of efficiency and efficacy. It is worth mentioning that it is the first step to identify the environmental properties and it supports an interaction between the environment and the learning system. The current work is an attempt to elaborate view into the existing knowledge representation in LCS domain in general and XCS in specific. The knowledge representation techniques are grouped into different categories based on the classification approach in which they are incorporated. In each category, the underlying rule representation schema, the format of classifier condition and a precise explanation on how it can partition the problem space along with the experimental results are provided. Additionally, to have an elaborated view on the functionality of each technique, a comparative analysis of existing techniques on some conventional problems is provided. It is worth mentioning that the main focus of current study is not explicitly on presenting an alternative knowledge representation scheme, rather it focuses on providing a guideline to choose a proper representation technique for a given problem which is nowadays human need. Undoubtedly, addressing the none-trivial recent issue is of great importance. We hope that this survey facilitates a better understanding of the different streams of research on this topic and provides a guideline for choosing a proper knowledge representation technique for a given problem. Furthermore, it can be of interest to the LCS researchers and practitioners and also opens up new streams of research on this topic.

The rest of this paper is organized as follows; at first in Section 2, we briefly introduce learning classifier systems and provide a description of XCS in detail. Section 3 describes the knowledge representation component and the various works that have been done to enhance this component. We group knowledge representation techniques into different categories based on the classification approach in which they are incorporated. In the subsections of Section 3, the underlying rule representation schema, the format of classifier condition and a precise explanation on how it can partition the problem space along with the experimental results are provided. In Section 4, to elucidate the functionalities of each technique a comparative analysis of existing techniques on some conventional problems is provided in general perspective. The last section includes summary and conclusion remarks.

2 Learning Classifier Systems in Brief

Learning Classifier Systems belong to a family of machine learning techniques which combines genetic algorithm (GA) with the power of the reinforcement learning paradigm to solve a given problem. In Holland's recipe, as the inventor of LCS, it has four main components with specific goals: (1) a population of classifiers known as knowledge representation component, which represents the current system knowledge; (2) a performance component, which provides a proper input for system and an applicable output to apply to the learning environment; (3) a reinforcement (or credit assignment) component, which is responsible to distribute the incoming reward among the classifiers that are accountable for it; (4) a rule discovery component, which creates new rules through a covering mechanism and evolves the existing ones by using an evolutionary algorithm, usually a GA. Most of the developed models which had been proposed in the realm of Michigan style LCS were extended and improved the original idea of Holland's LCS, but kept these entire four main components. It is notable that the effectiveness of LCS to solve a problem depends on the proper interaction between these four components, therefore the compatibility between them must be considered in developing a new extension of LCS.

In each learning step of a Michigan style LCS, the performance component perceives the environmental state (or input) through its detectors and performs the selected action through its effectors. Then, the environment eventually pays a reward to the system regarding the selected action's efficacy. Like other reinforcement learning techniques, LCS tries to suggest a solution which can maximize the amount of received reward through, a set of condition-action-prediction rules called classifiers which are produced and evolved to represent the current solution. On the other hand, the reinforcement component acts on the population to estimate the action values in each subproblem. The discovery component produces new rules for unseen subproblems and usually uses genetic algorithm to evolve the current solution. In the next section, a brief review on XCS (a milestone of LCS) and its main components is provided.

2.1 XCS in brief

The year 1995 is marked a milestone in LCS researches due to the appearance of the most successful and popular Michigan style LCS named XCS. Recent analysis and researches (Wong and Schulenburg, 2007; Bull et al., 2008; Studley and Bull, 2005; Vargas et al., 2004) have shown that XCS has an especial architecture that makes it a competitive learning system in solving complex problems. Its architecture ingredients provide some promising properties such as its online learning capability, its noise robustness, the generality in the learning mechanism, and its continuous adaptation.

XCS contains a population of classifiers which is called [P]. This population can be empty in the beginning of the experiment, or be filled randomly. Each classifier in [P] is made up of different parts. These parts are: a condition usually from the alphabet {0, 1, #}, an action which is usually an integer, and a set of associated parameters. These parameters are (1) a payoff prediction P_j, which estimates the payoff that the system will receive when its action is applied to the corresponding environment; (2) the prediction error ϵ_j, (3) the fitness F_j, and some other parameters such as *exp*, *num* and etc.

When XCS receives an environmental state, it forms the related match set [M]. This set includes those classifiers whose condition parts match the current environmental state. If no classifier matches, the covering operator will create a predefined number of classifiers which match the current input and insert them into the population and into [M]. If the covering operator causes the size of the population to grow over a predefined threshold *N*, some other classifiers will be eliminated from the population regarding their fitness and experience (*exp* parameter).

Then, for each action a_k, which is proposed by classifiers in $[M]$, the system computes a fitness weighted average P_k using this equation: $P_k = \frac{\sum_{cl\in[M|a_k]} cl^k.F \times cl^k.P}{\sum_{cl\in[M]} cl^k.F}$ where $[M|a_k]$ is the subset of $[M]$ contains classifiers which propose the action a_k.This value is used as the bid of the corresponding action to win the current phase. Then, XCS chooses an action from those proposed in $[M]$ regarding its Explore/Exploit strategy. Finally, an action set $[A]$ is formed which consists of a subset of $[M]$ with the same action as the chosen one.

After that, the selected action is applied to the environment and a reward R is received from the environment which may be zero. Then, the parameters of the involved classifiers are updated. In the sequential environments, the update procedure occurs for classifiers in the previous action set which is called hereafter $[A]^{-1}$. It is also notable that all updates are done using a Q-Learning update regime. To do so, the payoff P is calculated as follows: $P = r_{-1} + \gamma \max_a p_a$ where r_{-1} is the previous environmental reward which is 0 or 1000 for classification problem, γ is the discount factor and pa is the predicted payoff for the applied action in the current trial. Then, classifier predictions in $[A]^{-1}$ are updated as follows: $P_j = P_j + \beta(R - P_j)$ where β is the learning rate in the Widrow-Hoff update rule. Next, the prediction errors are re-estimated using this equation:$\epsilon_j = \epsilon_j + \beta(|R - P_j|)$. Then the accuracy for the corresponding classifier is calculated as follows: $k_j = 0.1(\frac{\epsilon_j}{\epsilon_0})^{-\nu}$ for $\epsilon_j > \epsilon_0$, else 1.0. The parameter ϵ_0 is termed the error threshold and ν is a positive integer and both of them are initiated at the beginning of the experiments. Then, the relative accuracy of each classifier is calculated as: $k'_j = \frac{k_j}{\sum_j k_j}$. And at last, this relative accuracy is used to update the classifier fitness: $F_j = F_j + \beta(|k'_j - F_j|)$. These updated values are used in another important component of XCS: The discovery component. On a predefined period based on the parameter θ_{ga}, a GA is applied to $[A]^{-1}$. As usual, applying the GA consists of three phases: the selection, the crossover and the mutation. In the selection phase, two classifiers are selected with the proportionate selection operator regarding their fitness. The crossover operator is applied to the two selected parents at the rate of χ. Then, at the rate of μ, each allele of the generated offspring is mutated. The resulting offspring are inserted into the population. If the size of the population grows over a predefined threshold *N*, two classifiers will be deleted from the population considering their fitness.

To improve the generalization capability of XCS, another concept is utilized: the subsumption deletion. When a classifier is added to the population by GA, it is checked against all classifiers in the population to find a sufficiently experienced and accurate classifier which covers the newly inserted classifier. If the covering classifier is found, then the new classifier will be eliminated from the population and the numerosity parameter (*num*) of the covering classifier will be added by one. This process is called GA subsumption. A same process also occurs in the action set creation phase. In this phase, all classifiers in $[A]$ are matched against accurate and sufficiently experienced classifiers in $[A]$. If a classifier is covered by another classifier, it will be eliminated from the population and the numerosity parameter of the covering classifier will be increased by one.

Figure 1 gives an overall picture of XCS system, which is shown in interaction with an environment. Above XCS description is an abbreviate one, for more details; see the original XCS paper (Wilson, 1995) and (Butz and Wilson, 2001).

3 Knowledge Representation

The first and main step to solve a problem is finding the proper realization of its definition and input space. In the other words, the solver system must be able to model the regularities of the problem space to make decision regarding its experiences. In rule based systems, their rule set is the one responsible for such modeling where each rule represents some portion of the problem space and makes the best decision in corresponding subspace. Consequently, a rule based system is able to solve a problem efficiently if the rule set can cover the whole problem space properly and also each rule makes effective decision. Therefore, it can be concluded that rule representation, i.e. knowledge representation,

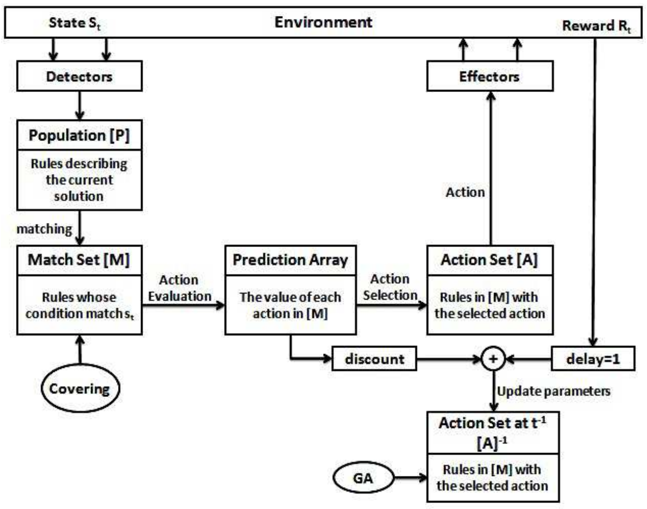

State S_t
Environment
Reward R_t
Detectors
Effectors
Population [P]
Rules describing the current solution
Action
matching
Match Set [M]
Rules whose condition match s_t
Action Evaluation
Prediction Array
The value of each action in [M]
Action Selection
Action Set [A]
Rules in [M] with the selected action
discount
+
delay=1
Covering
Update parameters
Action Set at t^{-1} $[A]^{-1}$
GA
Rules in [M] with the selected action

Fig. 1 Schematic illustration of XCS life cycle. In a typical iteration, XCS receives the current environmental state then forming $[M]$ from $[P]$. Next, the proper action is chosen to execute in the environment. The rule parameters would be updated according to the resulting reward. Finally, the GA may be applied on $[A]$.

has an essential role in rule based systems. In particular, a rule based classifier like XCS can achieve all the three main objectives of classification task which are high accuracy, comprehensibility and compactness, provided that it has a proper rule representation.

The first and main step to solve a problem is finding the proper realization of its definition and input space. In the other words, the solver system must be able to model the regularities of the problem space to make decision regarding its experiences. In rule based systems, their rule set is the one responsible for such modeling where each rule represents some portion of the problem space and makes the best decision in corresponding subspace. Consequently, a rule based system is able to solve a problem efficiently if the rule set can cover the whole problem space properly and also each rule makes effective decision. Therefore, it can be concluded that rule representation, i.e. knowledge representation, has an essential role in rule based systems. In particular, a rule based classifier like XCS can achieve all the three main objectives of classification task which are high accuracy, comprehensibility and compactness, provided that it has a proper rule representation.

As mentioned earlier, knowledge representation is an essential component of XCS whose elements are certain number of classifiers. Each classifier has two main parts; a condition which represents the regularities of the problem regarding effective generalization, and, an action part that shows what decision being made in corresponding condition. These parts can be adapted for a particular purpose, without modifying the main structure of the system.

In the problems with binary inputs, the first and most commonly used syntax for classifier condition termed "Ternary" can be used as done in (Wilson, 1995; Holland and Reitman,1978; Schuurmans and Schaeffer, 1989). In this syntax, the condition part is simply represented by a fixed length bit string defined over the alphabet {0, 1, #} where the *'don't care'* symbol (#) matches both one and zero. It is mentioned in (Schuurmans and Schaeffer, 1989) that *"Ternary representation can hardly model the relation among problem and there is an avoidable bias in system generalization capabilities"*, it has three noticeable advantages; first, it is easy to analyze and the obtained rule set is well understood; second, it is proper for textual and categorical information; and third, any data are eventually changed into binary in a computer system. However, since most of real world data sets contain real valued, nominal or mixed attributed data, researchers encourage to proposed powerful representation which are applicable to the complex problems with such data sets.

Many representations have been developed to handle real valued data, such as, disjunctions of intervals (Wilson, 2000a; Wilson, 2000b; Stone and Bull, 2003; Dam et al., 2005a), ellipsoidal (Butz, 2005; Butz et al., 2006; Butz et al., 2008) and convex hulls (Lanzi and Wilson, 2006). Other general purpose representations have also been developed, such as, first order logic (Mellor, 2005; Mellor, 2006), fuzzy logic (Valenzuela-Rendón, 1991; Bonarini and Matteucci, 2007; Casillas et al., 2007; Orriols-Puig et al., 2008a), GP-like conditions (Lanzi and Perrucci, 1999; Lanzi, 2001; Lanzi, 2003; Wilson 2008; Preen and Bull, 2009), messy coding (Lanzi, 1999), tile coding (Lanzi et al. 2006) and neural network (Bull and O'Hara, 2002; Hurst and Bull, 2004; O'Hara and Bull, 2005; Howard et al., 2008; Dam et al., 2008) based representation. Table 1 shows the synopsis of different kinds of knowledge representation emerged in XCS. The next subsections arranged in such a way that they detail knowledge representation schemes provided in Table 1 respectively.

Further on, we group the knowledge representation techniques into different categories described in each subsection based on the classification approach in which they are incorporated. In each category, we present the following issues: the underlying rule representation schema and the format of classifier condition to support the corresponding representation, a precise explanation bringing insight into how to partition the problem space and the extensive experimental results reported in the original papers.

Table 1 A summary of different rule representation methods developed for XCS and some of their features, such as; LCS algorithms/platforms, the year of proposing corresponding representation, and what articles have used such representation.

Rule Representation		System	Introduced in	Used in
Interval Based	Center Spread Representation	XCSR	Wilson, 2000a	Wilson, 2000a;
Representation	Lower-Upper Bound Representation	XCSI	Wilson, 2000b	Wilson, 2001b;
	Unordered-Bound Representation	—	Stone and Bull, 2003	Wilson, 2001;
	Min-Percentage Representation	—	Dam et al., 2005	Bernadó-Mansilla et al., 2001;
				Wilson, 2002;
				Hurst and Bull, 2004;
				Gao et al., 2005;
				Dam et al., 2005b;
				Hamzeh and Rahmani, 2005;
				Gao et al., 2006;
				Gao et al., 2007;
				Hamzeh and Rahmani, 2007;
				Gao et al., 2007;
				Llorá et al., 2007;
				Lanzi and Loiacono, 2007
Ellipsoidal Based	Hyper sphere Representation	—	Butz, 2005	Butz et al., 2005;
Representation	Hyper ellipsoidal	—	Butz, 2005	Butz et al., 2006;
	General hyper ellipsoidal	—	Butz, 2005	Wilson, 2001; Butz et al., 2008
Convex Hull Based Representation		—	Lanzi and Wilson, 2006	Lanzi and Wilson, 2006;
Fuzzy Logic Based		Fuzzy LCS	Valenzuela-Rendón, 1991	Valenzuela-Rendón, 1991
Representation		ELF	Bonarini, 1993	Bonarini, 1993;
		LFCS	Bonarini, 2000;	Bonarini, 1994;
		LFCS	Bonarini et al., 2000	Bonarini, 1998;
		FIXCS	Bonarini and Matteucci, 2007	Walter and Mohan, 2000;
		Fuzzy-XCS	Casillas et al., 2004	Bonarini, 2000;
		Fuzzy-UCS	Orriols-Puig , 2007	Bonarini et al., 2000;
				Casillas et al., 2004;
				Casillas et al., 2005;
				Casillas et al., 2007;
				Orriols-Puig et al., 2007;
				Bonarini and Matteucci, 2007;
				Orriols-Puig et al., 2008a;
				Orriols-Puig et al., 2008b
First Order Logic Based		FOXCS	Mellor, 2005	Mellor, 2005;
Representation				Mellor, 2006;
				Mellor, 2008;
Messy Code Representation		XCSm	Lanzi, 1999	Lanzi, 1999;
GP-like	S-expression	XCSL	Lanzi and Perrucci, 1999	Lanzi and Perrucci, 1999;
Representation	Stack Based Representation	—	Lanzi, 2003	Lanzi, 2001a;
	(PRN expression)			Lanzi, 2001b;
	GEP	XCSF-GEP	Wilson 2008	Lanzi, 2003;;
	Dynamical Genetic Programming	DGP-XCS	Preen and Bull, 2009	Unold, 2005
	(graph based representation)			Unold and Cielecki, 2005;
				Unold, 2007;
				Cielecki and Unold, 2007;
				Wilson,2008;
				Preen and Bull, 2009
Neural Networks Based	Neural Network	X-NCS	Bull and O'Hara, 2002	Bull and O'Hara, 2002;
Representation		NCS	Hurst and Bull, 2004	Hurst and Bull, 2004;
		NLCS	Dam et al., 2008	O'Hara and Bull, 2005;
		XCSFNN	Loiacono and Lanzi, 2006	Loiacono and Lanzi, 2006;
	Fuzzy-Neural Network	X-NFCS	Bull and O'Hara, 2002	Dam et al., 2008
Tile Coding Based Representation		XCSF-RTC	Lanzi et al., 2006	Lanzi et al., 2006;

3.1 Interval Based Representation

The traditional ternary representation has been substituted with interval-based representation to manage continuous-valued inputs. It has been shown that this modified XCS can be effectively applied to real-valued problems (Bernadó-Mansilla et al., 2001; Wilson, 2000b; Wyatt, 2004; Bull et al., 2008). In the interval-based representation condition part of each classifier is determined using intervals defined over each dimension. To define these intervals, four representation techniques have been introduced as follows:

Center Spread Representation (CSR): In (Wilson, 2000a), Wilson modified XCS and introduced CSR to handle real-valued inputs. In CSR, for the solution space of range $[p_{min}, q_{max})$, an interval predicate $[p_i, q_i)_p$ is represented as a tuple $(c_i, s_i)_g$ where c_i indicates the center of the interval and s_i is a radius around c_i.

Both c_i and s_i are real valued numbers. So, a classifier matches input x if and only if $c_i - s_i < x_i < c_i + s_i$ for all dimension of x.

Stone and Bull (Stone and Bull, 2003) analyzed the behavior of classifiers with this representation in XCS. They argued that using CSR causes the incomplete and many to one genotype to phenotype mapping. As an example, the possible values of i'th dimension vary between [0,10), the interval [7,10) in phenotype space can be mapped by (9,2),(8.5,1.5), (10,3), and many more.

Once nonlinear genotype to phenotype mapping has performed and mutation and crossover operators have applied, a truncation operator must be applied to classifiers to keep them in range. This operator introduces a bias in the distribution of intervals phonotype. Through this bias, the frequencies of $[p_{min}, q_i)_p$ and $[p_i, q_{max})_p$ intervals increase as q_i increases or p_i decreases, respectively. Therefore, the $[p_{min}, q_{max})_p$ interval would have much greater chance to be appeared in classifier conditions.

Lower-Upper Bound Representation or Min-Max Representation (MMR): Wilson lately proposed another representation named MMR (Wilson, 2000b) for integer-valued inputs. It can also be used for real-valued inputs (Wilson, 2002). Here, an interval predicate $[p_i, q_i)_p$ is encoded as a tuple $[l_i, u_i)_g$ where $l_i = p_i$ and $u_i = q_i$. In real valued problems l_i,$u_i \in R$ and in integer-valued problems $l_i, u_i \in Z$. A classifier matches input x if and only if $l_i < x_i < u_i$ for all dimension of x.

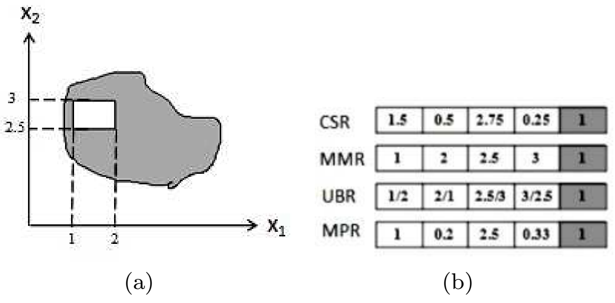
Stone and Bull (Stone and Bull, 2003) stated that this representation has complete and one to one genotype to phenotype mapping. Furthermore, this method overcomes the bias generated by truncation in CSR. In comparison with CSR, in MMR, the frequency of intervals of the form $[p_{min}, q_i)_p$ and $[p_i, q_{max})_p$ is constant for all p_i and q_i and $[p_{min}, q_{max})_p$ interval has same frequency as that of any other interval. But, another issue arises; the tuples have an ordering restriction ($l_i < u_i$) and genetic operators might produce infeasible intervals.

Unordered-Bound Representation (UBR): In order to overcome the ordering issue of the previous method, Stone and Bull (Stone and Bull, 2003) proposed a new approach named unordered-bound representation (UBR) which is similar to MMR without ordering restriction. Thus, an interval predicate $[p_i, q_i)_p$ can be defined in the form of $[l_i, u_i)_g$ or $[l_i, u_i)_g$ where $l_i \neq u_i$ and $l_i = p_i$ and $u_i = q_i$. In other words, the genotype to phenotype mapping is normally complete and two to one except where $q_i = p_i$ which is one to one. The mutation operator provides strong specialization pressure with UBR, unlike MMR and CSR, where the mutation operator yields a low level of specialization pressure.

Although UBR is able to obviate the problems of previous approaches, it raises new issues. The semantic of the genotypes is not permanent; that is, the gene presenting lower bound of interval in one generation may present the upper bound in the next generation. This issue is inconsistent with building block hypothesis which is the keystone in the architecture of GA and is known as one of the important necessities to run it (Goldberg, 1989).

Min-Percentage Representation (MPR): In (Dam et al., 2005a), Dam et al. purposed to overcome the problem of UBR by presenting new approach named Min-Percentage Representation which maintains the semantic of the genotype all over evolutionary run. In MPR, an interval predicate $[p_i, q_i)_p$ can be encoded as a tuple $[m_i, per_i)_g$ where $m_i = p_i$ and $per_i = \frac{q_i - p_i}{p_{max} - p_i}$. So a classifier matches input x if and only if $m_i < x_i < m_i + per_i * (p_{max} - m_i)$ for all dimension of x.

In (Dam et al., 2005a), MRP and UBR were compared in 6-Real-Multiplexer and Checkerboard problems. Results indicated that the two techniques have equivalent performance and similar behavior in both problems. But, MRP is unlikely to change the semantic of the genotypes.



CSR	1.5	0.5	2.75	0.25	1
MMR	1	2	2.5	3	1
UBR	1/2	2/1	2.5/3	3/2.5	1
MPR	1	0.2	2.5	0.33	1

Fig. 2 An example of interval based classifier in an arbitrary problem space. a) Visualization of the covered area by interval based classifier. b) The genotype of the classifier presented in (a) with four different types of interval based representation. The white cells identify the condition part and the gray cell identifies the action part of the corresponding classifier.

As illustrated in Figure 2, XCS with interval based representation uses hyper rectangles to partition the problem space. In (Butz et al., 2006) Butz showed, that the interval based condition makes difficulties to solve problems with oblique decision boundaries. He also showed that the learning takes longer time and the compactness of the model is not efficient when dealing with oblique boundaries. For example, XCSF (Wilson, 2002), one of the successful extensions of XCS, needs an enormous number of classifiers with hyper rectangle conditions to accurately approximate a nonlinear surface such as a circle. Interval based representation is well-suited in modeling axis-parallel boundaries, but in dealing with oblique boundaries, it is more advantageous to use a representation which partitions the problem space into complex regions.

3.2 Ellipsoidal Based Representation

In (Butz, 2005), Butz suggested a new structure of classifier condition, based on defining hyper sphere and hyper ellipsoid shapes. There, three structures were proposed; first, the condition part of the classifiers represents a hyper sphere in the problem space. As hyper sphere has common radius in all dimensions, the condition part consists of a center and a deviation, that is, $C = \{\overrightarrow{m}, \sigma\} = \{m_1, m_2, ..., m_n, \sigma\}$. Second, the condition part represents an axis-parallel hyper ellipsoid which has different deviation in each dimension, that is, $C = \{\overrightarrow{m}, \overrightarrow{\sigma}\} = \{m_1, m_2, ..., m_n, \sigma_1, \sigma_2, ..., \sigma_n\}$. Third, the condition part is redefined to present a general hyper ellipsoidal structure which is axis-independent unlike the previous one. So, it is defined by a center point in addition to elements of matrix Σ named transformation matrix which indicates fully Mahalanobis distance metric of a hyper ellipsoid. This matrix shows stretch and rotation of the represented hyper ellipsoid. The condition part of a classifier is defined as follows: $C = \{\overrightarrow{m}, \Sigma\} = \{m_1, m_2, ..., m_n, \sigma_{1,1}, \sigma_{1,2}, ..., \sigma_{(n,n-1)}, \sigma_{(n,n)}\}$ where $\overrightarrow{m}$ shows the center of represented hyper ellipsoid and Σ shows the transformation matrix of the condition. This structure is a general form of the first and second ones; hyper sphere is a hyper ellipsoid where the diagonal entries of Σ are initialized with common value and all other entries are set to zero, and axis-parallel hyper ellipsoid has a diagonal matrix as Σ. In the general form, angular orientation and stretch of the represented hyper ellipsoid are implicitly encoded in Σ. Due to the redundancy of this encoding; the mutation and crossover operators may not act beneficially (Butz et al., 2008). This would decrease the reproductive opportunities of the successful classifiers and lead the evolutionary progress to slow down. To solve this problem, Butz et al. (Butz et al., 2006; Butz et al., 2008) investigated another condition representation which is able to explicitly codify the rotation of the desired hyper ellipsoid in its structure. So, the condition part of each classifier is expressed by three vectors, $C = \{\overrightarrow{m}, \overrightarrow{\sigma}, \overrightarrow{\gamma}\}$; a vector $\overrightarrow{m}$ that indicates the center point, a vector $\overrightarrow{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_n)^T$ which represent the stretch, and a vector $\overrightarrow{\gamma}$ with size of $\binom{n}{2}$ to point out the orientation angles of the corresponding hyper ellipsoid. Figure 3 highlights how a classifier with mentioned representations can cover a partition of the input space.

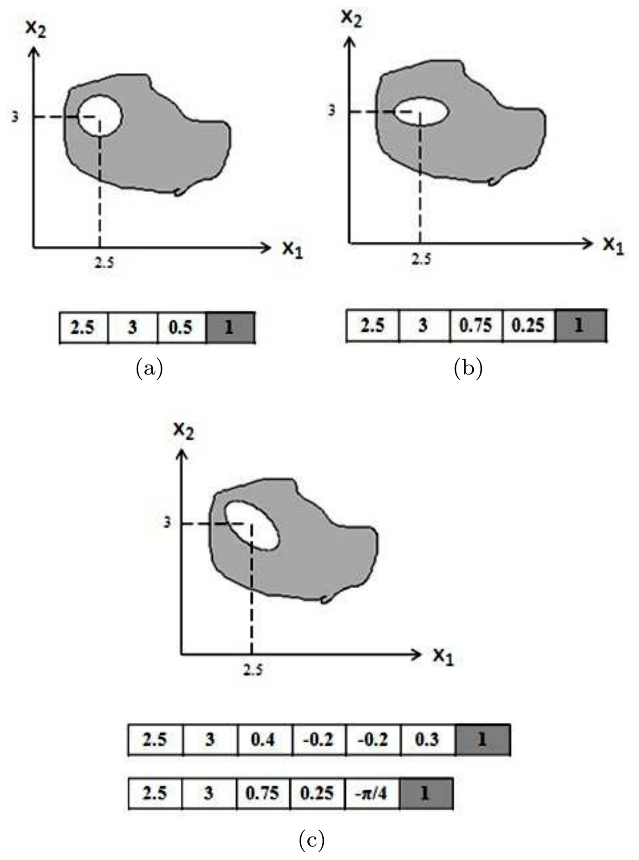


Fig. 3 An example of the ellipsoidal based classifier in an arbitrary problem space. a) Visualization of the covered area by a hyper spherical classifier and its genotype. b) Visualization of the covered area by an axis-parallel hyper ellipsoidal classifier and its genotype. c) Visualization of the covered area by a general hyper ellipsoidal classifier and its genotypes in two cases; in the upper one, matrix Σ is used to identify the rotation of the represented ellipsoid and in underneath case, the orientation angle is presented. The white cells identify the condition part and the gray cell identifies the action part of the corresponding classifier.

In all these structures, activation of each classifier in forming $[M]$ is determined by a Gaussian kernel function, applied to the distance between the current input and the center point. So, to find whether a classifier can match the current input x or not, at first, the activation of the classifier $cl.ac$ is computed using Formula 1, 2 or 3 according to its structure. Then, the current input will be matched with a classifier if the activation of such classifier is greater than a threshold θ_m. Formula 1, 2 or 3 are used when the condition of the classifier represents a hyper sphere ($C = \{\overrightarrow{m}, \sigma)\}$, an axis-parallel hyper ellipsoid ($C = \{\overrightarrow{m}, \overrightarrow{\sigma}\}$) , or a general hyper ellipsoid ($C = \{\overrightarrow{m}, \Sigma\}$) respectively.

$$cl.ac = exp(-\frac{\|x-m\|^2}{(2\sigma^2)}). \quad (1)$$

$$cl.ac = exp(-\sum_{(} i = 1)^n \frac{(x_i - m_i)^2}{(2\sigma^2)}). \quad (2)$$

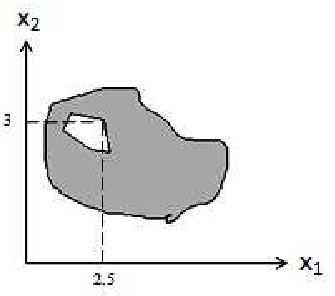
$$cl.ac = exp(-\frac{(\overrightarrow{x} - \overrightarrow{m})^T \Sigma^T \Sigma (\overrightarrow{x} - \overrightarrow{m})}{2}). \quad (3)$$

Promising results of these condition representation approaches reported in (Butz, 2005; Butz et al., 2006; Butz et al., 2008) showed an improvement in performance of XCS (or XCSF) in continues space problems. Especially, it is more profitable to use a more general condition structure while the dimensional dependencies of the problem are unknown. Although these representations can appropriately model oblique boundaries, but it must be noted that it might be not well-suited where decision boundaries of problem are axis-parallel for which case interval based representation can be applied effectively.

3.3 Convex Hull Based Representation

Another approach that can be utilized to represent the condition of a classifier in the real-valued problems is using the convex hull concept as proposed in (Lanzi and Wilson, 2006). In this approach, condition part of each classifier comprises a set of points in the problem space that identifies a convex hull. In other words, each classifier depicts a convex region of the problem space and matches all problem instances which lie inside this region, as shown in Figure 4. As all the other geometric shapes can be approximated through a convex hull with sufficient number of points, the convex hull based representation is a general form of both previous representations. Besides, the convex hull representation has a fine ability to identify more complex regions due to its asymmetric shape.

Condition part of the classifiers can represent a convex hull in two manners; first, using a set of points. Second, each classifier can define a convex hull directly by presenting angles and radius of the convex hull in its condition part, instead of using a set of points. In both cases, number of convex hull vertices can be fixed or variable. In (Lanzi and Wilson, 2006), the authors discussed the influence of variable sized conditions and concluded that *' represent arbitrarily complex convex regions but increase the complexity of the genetic search and also introduce the bloating phenomena that are typical with variable size representations '*.



(a)

2.5	3	2.7	2.4	2	2.5	1.3	2.8	1.5	3.2	1

(b)

Fig. 4 An example of a convex hull based classifier in an arbitrary problem space. a) Visualization of the covered area by a convex hull based classifier. b) The genotype of a classifier presented in (a) which defines the points to represent the desired convex hull. The white cells identify the condition part and the gray cell identifies the action part of the corresponding classifier.

In (Lanzi and Wilson, 2006), the performance of XCSF based on convex hull representation was compared to the XCSF with interval-based representation. Results showed the fast convergence of system when the problem space is partitioned by convex regions. Also it was shown that XCSF with variable sized condition converges faster than a version of XCSF where condition part of classifiers consists of 10 or 15 points, but compared to 3 and 5 points, it converges slower.

3.4 Fuzzy Logic Based Representation

In the rule based classifier systems, the comprehensibility and interpretability are two must considerable features (Hayes-Roth, 1985). Besides, fuzzy logic is one of the best known mechanisms providing such properties. There are number of approaches to use fuzzy logic and fuzzy set theory (Zadeh, 1965; Zadeh, 1973) as a technique for representing rules in Michigan-style LCSs, such as (Valenzuela-Rendón, 1991; Bonarini and Matteucci, 2007; Casillas et al., 2007; Orriols-Puig et al., 2008a). The main goal behind such efforts is combining the generalization capabilities of LCS with the fine interpretability of fuzzy rules to achieve an online learning system with more accurate, general and well understandable rule set. In the following, first, we briefly describe important works in this area. Second, a comprehensive description of notable approaches which try to embed fuzzy logic in knowledge representation component of LCS is provided.

Early attempts to integrate fuzzy logic and learning classifier systems were proposed in (Valenzuela-Rendón, 1991; Nomura et al., 1998; Parodi and Bonelli, 1993). Valenzuela-Rendón in (Valenzuela-Rendón, 1991) introduced the first proposal of Learning Fuzzy-Classifier Systems (LFCSs), which is a Michigan style LCS and consists of fixed-size fuzzy-rule set and a fuzzy message list. Afterwards, several researchers have used the idea of fuzzy logic or fuzzy set theory into LCS to have an online fuzzy rule based system which can be used in many tasks (Nomura et al., 1998; Parodi and Bonelli, 1993; Furuhashi et al., 1994; Nakaoka et al., 1994; Velasco, 1998; Ishibuchi et al., 1999). Many of proposed systems were applied to the reinforcement learning and control tasks. As the initial LFCS framework was not completely coincident with reinforcement learning architecture, Nomura et al. (Nomura et al., 1998) improved the LFCS structure to be a true reinforcement learning technique. In (Parodi and Bonelli, 1993), Parodi and Bonelli defined an LFCS which can automatically learn the fuzzy relations, membership function and weights. Velasco (Velasco, 1998) designed a new extension of LFCS especially designed for fuzzy process controls and Ishibuchi et al. (Ishibuchi et al., 1999) proposed an LFCS for pattern classification.

The research on this topic had been continued by the works of Bonarini (Bonarini et al., 2000; Bonarini, 1998; Bonarini, 2000; Bonarini and Matteucci, 2007). He addressed the classic "Competition versus Cooperation" problem in genetic fuzzy systems, (Bonarini, 1996; Bonarini and Trianni, 2001). He proposed a Michigan style LCS named ELF where the rule set is divided into subpopulations. To produce the correct action, the classifiers of these subpopulations cooperated whilst the classifiers in each subpopulation competed with each other. The behavior of ELF in overcoming some of issues of strength-based LCSs was verified by applying it to several reinforcement learning problems such as the coordination of autonomous agents. In (Bonarini, 2000; Bonarini et al., 2007), a general framework of learning classifier systems were introduced and later this framework was extended particularly for XCS called FIXCS in (Bonarini and Matteucci, 2007). In (Bonarini, 2000), the different components of this framework have been analyzed to be consistent with fuzzy models. In addition, some features are introduced for the sake of classifying LFCS proposals presented in the literature.

Recently, in (Casillas et al., 2004; Casillas et al., 2007) Casillas et al. also used fuzzy logic in knowledge representation component of XCS to express rules in fuzzy format. There, the theoretical issues of applying fuzzy model in accuracy based learning classifier systems have been investigated. The proposed approach was named Fuzzy-XCS which is the first successful accuracy based fuzzy rule based system with generalization ability. Soon after, the Orriols-Puig proposal (Orriols-Puig et al., 2008a; Orriols-Puig et al., 2008b), named Fuzzy-UCS, extended this approach to be applied in supervised learning classifier system, i.e. UCS which is a derivation of XCS introduced in 2003 (Bernadó-Mansilla and Garrell, 2003) for classification task in data mining. In following, a comprehensive description of well known fuzzy representation which is successfully used in LCS realms to produce accuracy based fuzzy rule based system such as Fuzzy-XCS and Fuzzy-UCS.

The main idea of using fuzzy logic in XCS as the knowledge representation tool is to represent the labels associated to fuzzy sets in the rule's structures and offer LCS a mechanism to evolve them. This mechanism must be consistent with fuzzy rules and it must also be able to learn a rule set in order to implement an input to output mapping where both input and output can be either real valued or nominal. In common manner, the rule set consists of fuzzy rules which are defined in disjunctive normal form (DNF) with the following structure:

$$\textbf{\textit{IF}}\ X_1\ \textit{is}\ _1\ \textit{and}\ ...\ \textit{and}\ X_n\ \textit{is}\ _n\ \textbf{\textit{THEN}}\ Y\ \textit{is}\ B. \qquad (4)$$

Where each input variable X_i is described through a set of linguistic terms, $_i = \{A_{i1}V...VA_{il}\}$, represented by a disjunction (T-conorm operator) of l linguistic terms. Meanwhile, each output variable is represented by a usual linguistic variable. For example, in (Casillas et al., 2007) to use this representation in classifiers of Fuzzy-XCS, a binary coding schema for the condition part and an integer coding schema for the action part were proposed. The number of bits considered in the condition part is equal to the total number of defined linguistic terms for all dimensions. For example, consider a problem with two dimensions where five linguistic terms are defined for both the first and second dimensions. So, the condition part of each classifier should have ten bits to codify these linguistic terms. The value of each bit indicates the appearance of the corresponding linguistic term. For each output variable, there is a gene in action part which denotes the index of utilized linguistic term. Below, there is an example to show how a rule can be codified by this representation:

Terms for each input/output variable:

vS [very small], S [small], M [medium],
L [large], vL [very large]

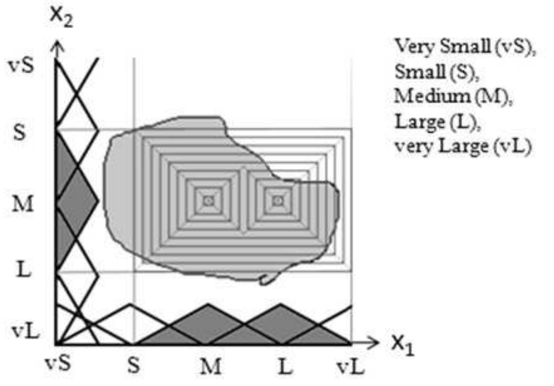
Fuzzy rule:

IF X_1 is {M,L} and X_2 is S THEN Y_1 is M and Y_2 is L

Representation:

[0011001000 || 23]

Figure 5 shows what portion of input space can be covered by a fuzzy classifier. As it is shown, each dimension is partitioned according to its defining linguistic terms and the covering region is denoted by a dashed rectangle.



(a)

Medium or Large					Small					2	3
0	0	1	1	0	0	1	0	0	0	2	3

(b)

Fig. 5 An example of a fuzzy classifier in an arbitrary problem space. a) Visualization of the covered area by fuzzy classifier. b) The phenotype and genotype of classifier presented in (a). The white cells identify the condition part and the gray cells identify the action part of the corresponding classifier

Rules with fuzzy structure entail several features making it useful as claimed in (Casillas et al., 2007):

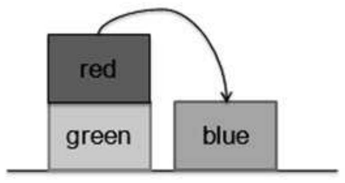
- It offers an approach to handle mixed attribute.
- It can handle missing values due to the natural capability of fuzzy logic in supporting the absence of some input variables.
- This structure leads to have more compact description and the evolved rule set consists of fuzzy rules with different generalization degrees.
- It is robust to noisy input.
- It can produce the output classes with a certainty factor in classification problems.

Promising results are obtained by evaluating the system on some function approximation problems and a realistic robot simulation online learning and supervised learning problems. Results show that it can increase the compactness of evolved rule set which is accurate, general and co-adapted However, using fixed and predefined fuzzy sets cause a limited number of possible rules. So, they are not flexible enough to fit the data very well. In order to overcome this issue, system must have an ability to modify the predefined fuzzy sets. In (Marin-Bl*á*zquez et al., 2007; Marin-Bl*á*zquez and Shen, 2008; Marin-Bl*á*zquez and Martnez Prez, 2008), linguistic hedges are employed to produce a new fuzzy set by modifying the original fuzzy set in interpretable manner where the original fuzzy set remains unaltered. By using linguistic hedges, the model has more freedom in domain knowledge representation and extraction. The research has demonstrated that better granularity of linguistic fuzzy modeling are achieved by using linguistic hedges as a technique to modify fuzzy sets and the inclusion of hedges causes improvement in the accuracy of the resulting system.

3.5 First Order Logic Based Representation

Mellor proposal in (Mellor, 2005; Mellor, 2006) extended XCS to a new model named FOXCS where rules are represented based on the first-order logic. First-order logic is useful to improve the expressive power of XCS due to its ability to present the complex relationships among attributes of a task domain. The modified classifiers are in form of Horn clauses which consist of three parts; an action part, a condition part and a background part, that is *action* ← *condition,Background*. Condition part consists of a number of variables. Here, variables have a generalization role in XCS similar to the # symbol in the ternary representation. However, variables can also be placed in action part. But, as a classifier should be defined in Horn clause form, its action part must consist of just one variable, i.e. atom. The background part can be empty.

The main advantages of using the first order logic are; 1) it is facilitated to represent relational concepts which relate variables of action part to those of condition, like *(A ← ABB)* where *A* and *B* are variables of given problem. Obviously, the variable A which takes place in action part can also appear in condition part. To illustrate this property, there is an example for a *Blocks World* tasks in Figure 6. 2) Another advantage is that rules can contain background knowledge which is a feature of many inductive logic programming (ILP) and rational reinforcement learning (RRL) systems, and can be helpful to solve these tasks effectively. 3) Mellor [Mellor, 2008] claimed that FOXCS as a RRL system is general, model free and "tabula rasa" system. It is general due to no restriction in problem framework and "tabula rasa"[1] because the initial policy can be left unspecified. The Results reported in (Mellor, 2005; Mellor, 2006; Mellor, 2008) show that the extended XCS can reach near optimal solution in the *Poker* and *Blocks World* problems which are not solvable by traditional XCS. Experimental results demonstrated that FOXCS can learn the optimal or near optimal policies where accuracy is comparable to the accuracy of many ILP algorithms in solving standard RRL tasks.



(a)

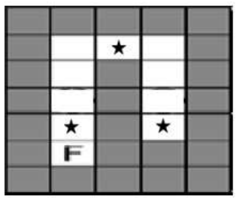
move(A,B) ← clear(A),clear(B), onFloor(C),onFloor(B),on(A,C)

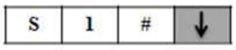
(b)

Fig. 6 An example of a first order logic based classifier in a Blocks World problem. a) Current situation of problem environment. b) A classifier based on first order logic, which can match the current situation. A, B and C are problem variables and $clear(X)$, $onFloor(X)$, $on(X,Y)$, and $move(X,Y)$ are functions. This classifier will match current situation if the A, B and C variables are assigned to *"red"*, *"blue"*, and *"green"* values.

3.6 Messy Code Representation

In (Lanzi, 1999), Lanzi introduced the messy version of XCS classifier system named XCSm in which the condition part of each classifier is defined by messy coding. The messy representation was firstly introduced by Goldberg in Messy Genetic Algorithm (Goldberg et al., 1989). In this representation, each gene is defined by a pair: *(Gene Number, Allele Value)*. As it is not necessary to have an allele for each possible gene in a chromosome, messy chromosome can be underspecified. In XCSm, a variable sized set of messy genes composes the classifier condition in which Gene Number presents the tested sensor and Allele Value is a fixed length bit string that specifies the sensed input. For example, consider a maze environment with eight sensors. The food, obstacle and empty cells are codified as *"11"*,*"10"*, and *"00"* alternatively. Any input state with a goal or obstacle in its south position is matched by a messy gene like (S,"1#"). Figure 7 shows an example of such classifier in Maze7 problem introduced in (Lanzi, 1998). As shown in Figure 7(a), the corresponding classifier can cover states marked by star in their center. With comparison to Ternary representation, the main advantageous of this representation is the independency of this representation's bits from the position of input sensors bits. Experimental results in (Lanzi, 1999) showed that





(a) (b)

Fig. 7 An example of a messy code based classifier in Maze7 (Lanzi, 1998) problem space. a) Visualization of the problem space. b) an arbitrary classifier. The matched states of environment states are marked in part (a) by a star in their center.

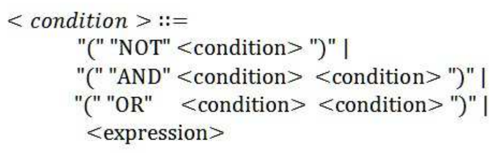
XCSm cannot reach the optimal solution. By analyzing the final population of classifiers, it is found that most of the evolved classifiers are over-general one. In order to cope with under-specification and over-specification of classifier condition, Lanzi developed XCSm by using high covering probability, extending mutation operator. He also suggested a new matching operator in which a condition matches current input if all its messy genes can match this input. As experiments notified, this new enhanced XCSm can learn an optimal solution.

3.7 GP-like Representation

The representation methods which represent a tree form as GP (Koza, 1992), an expression as BNF[2] grammar (Chomsky, 1956), and the other similar structures are known as GP-like representation. Firstly, using this representation in XCS was exploited by Lanzi in (Lanzi and Perrucci, 1999; Lanzi, 2001a). There, a system named XCSL was proposed that the condition parts of its classifiers have a general purpose representation, namely lisp s-expressions. The conditions were defined by the BNF grammar as shown in Figure 8. The non-terminal symbol <expression>in BNF form must be defined in

[1] Tabula rasa is the epistemological thesis that individuals are born without built-in mental content and that their knowledge comes from experience and perception.

[2] Backus Normal Form or Backus-Naur Form



```
< condition > ::=
        "(" "NOT" <condition> ")" |
        "(" "AND" <condition>  <condition> ")" |
        "(" "OR"    <condition>  <condition> ")" |
         <expression>
```

Fig. 8 The BNF grammar generating the overall structure of classifier conditions.

order to use this lisp like representation for a given problem.

To evaluate a classifier in the matching phase, the input data of XCSL must be represented as a string of attribute-value pairs and the terminal symbols presented in the classifier must be replaced with their actual values of corresponding attributes. Since the classifiers in XCSL have GP like structure, the genetic operators work as they do in traditional Genetic Programming (Koza, 1992).

As shown in (Lanzi, 2001b), XCSL tends to produce many overgeneral classifiers in population, so the learning performance of XCSL would be influenced in some cases. Besides, due to the bloat phenomenon (Langton et al., 1996; Soule et al., 1996), which is common in any GP like system, it is almost impossible to analyze what kind of classification model is developed by XCSL. To overcome the former problem, a final condensation phase was added into the system to extract a minimal subset of classifiers as the final solution. It extracts a compact solution from many overlapping classifiers instead of extracting all best classifiers from the population. The promising results suggested that XCS with symbolic based representation might be an interesting approach to extract useful knowledge from data. In (Lanzi, 2003), an extension to XCS was proposed by adding a stack-based representation into the classifiers' condition which was a linear program expression in Reverse Polish Notation (RPN) form. So, each condition consists of a sequence of tokens; each token can be a constant, a function or a variable which can be assigned to the corresponding input attribute value. There, a stack-based genetic programming, introduced in (Perkis, 1994), is used to mutate and recombine the classifiers. The reported results are quite interesting. Since genetic operators do not take into account any information about the structure of the operators defined in RPN and their arity, syntactically incorrect conditions can be easily generated. Consequently, the search space of the feasible solution would be even more highly redundant in comparison with XCSL.

Wilson has explored the use of gene expression programming (GEP) within XCS (Wilson, 2008) named GEP-XCS. The main aim of using GEP translation in XCS was not only to fit the environmental regularities better than rectilinear but also to produce a well understood rule set. The condition part of each classifier in GEP-XCS is expressed with an expression tree which can be evaluated by initializing the tree's terminals with the corresponding input attributes values. So, in the process of forming $[M]$, the obtained value of applying the presented expression in each classifier on the current input is compared with a predetermined threshold. If the obtained value exceeds this threshold, corresponding classifier will be added to the $[M]$.

The most important feature of using GEP translation is that every chromosome generated in GA cycle is syntactically valid. In addition, the size of expression tree in GEP representation is limited by the fixed chromosome size therefore the bloat phenomenon which usually happens in GP does not occur. GEP has further important properties such as combining several genes into a single chromosome and the concurrent evolution ability which are disregarded in GEP-XCS. Figure 9 illustrates how a GP-like classifier can partition the problem space. In this example, the classifier presents an expression tree form shown in Figure 9 (b). According to the different types, namely using GP or GEP, to encode the phenotype of the classifier, the expression tree can present in two different ways shown in Figure 9 (b). As mentioned, the condition part of classifier in GEP based representation must have fixed size; here it is assumed that the length of condition part of each classifier must be fifteen. Hence, since the corresponding tree form only needs seven genes to be encoded, in GEP based representation the additional genes are initialized with terminal variables. But, in the GP representation the classifier can have variable length so the length of classifier condition part can be equal to the number of nodes of the expression tree. Adding GEP to XCS as a condition representation has three main advantages; (1) each classifier has functional condition, with fine ability to fit environmental regularities, (2) the applied genetic operators can be quite simple because they are dealing with linear chromosomes instead of expression trees. Also, there is no need to verify the validation of produced offspring because genetic operators applying on GEP chromosomes always produce valid chromosomes and (3) the most attractive property which makes GEP based representation more desirable than other GP-like representations, is fixed size of expression tree. The reported results showed that using such representation can achieve good performance and gives greater insight into the environment's regularities but it leads fairly slower evolution and the evolved rule set was not compact.

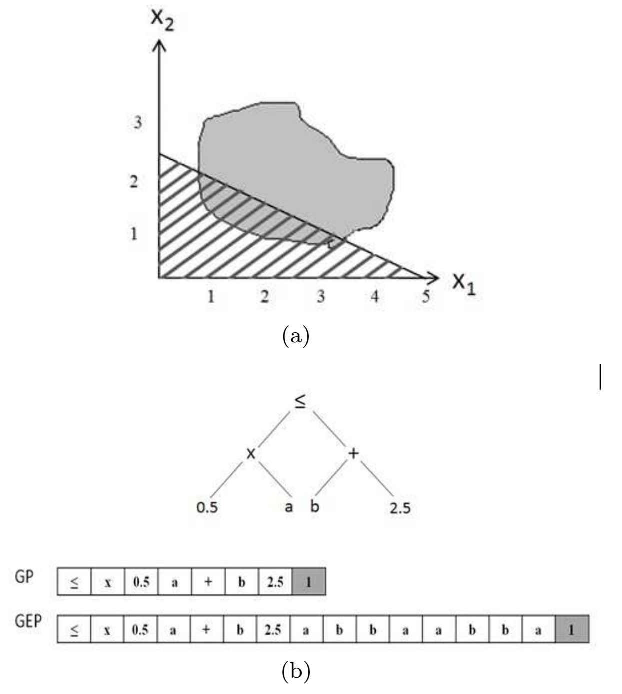


Fig. 9 An example of a GP-like classifier in an arbitrary problem space. a) Visualization of the covered area by a GP-like classifier. b) The phenotype and genotype of classifier presented in (a).

Recently, an investigation (Preen and Bull, 2009) was done in using a discrete dynamical system representation within XCS which is termed Dynamical Genetic Programming (DGP). DGP-XCS used a graph based representation for the condition of each classifier, wherein each node is a Boolean function. In other words, each classifier condition presents a Random Boolean Network (RBN) with N nodes which is equal to sum of the inputs I and the outputs O plus one ($N = I+O+1$). The first connection of each input node is assigned to the corresponding locus of such input and other connections are set at random. The condition part of each classifier presents a RBN with its true table and the connections of each node. This new system is tested on two most common benchmarks, the multiplexer and the maze navigation problem. According to the obtained results, it can be concluded that it is possible to design an ensemble of RBN by using XCS which is able to solve a computational task under RL scheme.

3.8 Neural Networks Based Representation

Larry Bull and Toby O'Hara proposed an accuracy based neural classifier named X-NCS and a Neuro-fuzzy classifier system called X-NFCS in (Bull and O'Hara, 2002). In X-NCS, both condition and action parts of each classifier present a single and full connected neural network like multilayer perceptron (MLP) (Bishop, 1995). In the condition part of each classifier, the weights of such small neural networks are concatenated together and can be evolved under GA mechanism in XCS. All the neural networks presented in all classifiers have the same number of nodes in their hidden layers. Besides the number of output nodes is equal to sum of the number of possible actions and one extra output node named *'not match-set member'*. Since all rules can see the input state, this extra output node is considered to signify whether the corresponding classifier is a member of the $[M]$ or not. Each classifier whose *'not match-set member'* node does not have the highest output value can be a member of the $[M]$. The winner action of each classifier is the one whose corresponding output node has the highest activation.

X-NCS had been tested on two tasks; a 6-bit multiplexer which is a single step task and to solve a maze named *Woods2* which is a multi step problem. The obtained results showed that X-NCS could solve both tasks. In addition, X-NCS components and parameters were modified to work as a function approximator. Its performance had been examined with root-mean-square. As expected, the results showed that the most accurate solution is achieved when one classifier can cover the whole input space. To extend X-NCS to Neuro-fuzzy version (X-NFCS), radial basis function (RBF) is used due to its similarity to the fuzzy rule based system (Jang and Sun, 1995). The discovery component in X-NFCS evolves RBFs as it does in X-NCS for MLPs.

One of the main advantages of this scheme, namely using neural network based representation, is its ability to be applied on problems with continuous action space. The promising results showed that both X-NCS and X-NFCS can be used in more complex tasks including both continuous and discrete action space and also it is applicable to both single step and multi step problems. To address the slow convergence in X-NCS, in (O'Hara and Bull, 2005) a hybrid technique was proposed to speed up the learning using Memetic Algorithm (MA) in XCS which augments the GA search with a local search (Moscato, 1989).

In (Howard et al., 2008), the principle of constructivism and self-adaptation was explored within XCS and XCSF. There, those systems were applied on well-known maze problems and the reported results showed using self adaptive operators, neural constructivism, and prediction computation could be an improvement in the realm of XCS. These systems can optimally perform in complex and noisy environments. Overall, the experiments of these mentioned papers and the similar ones showed that X-NCS is able to model the environment in simple robotics applications (Hurst and Bull, 2004) and also can solve several maze problems (Bull and O'Hara, 2002; O'Hara and Bull, 2005; Howard et al., 2008). It must be mentioned that the complete replacement of a rule by a neural network makes X-NCS to lose its main advantage as a rule based systems, that is, the potential ability to produce a well understood rule set. To overcome this issue, a new system named NLCS was proposed in (Dam et al., 2008). It is an extension of UCS which is driven for classification task in data mining problems. In (Dam et al., 2008), neural network is just used to identify the proper action of classifiers. The aim of such usage is to produce smaller evolved rule set while it still maintains or even improves the predictive accuracy. So, each classifier consists of two parts: the condition part which can be encoded by any existing representation method such as interval predicates (which is also used in (Dam et al., 2008)) like messy code and GP-like, and the action part which contains a neural network. For better understanding, Figure 10 shows the structure of a classifier in a two-dimension problem and how such classifiers will partition the problem space. As results showed (Dam et al., 2008), defining neural network in action part allows XCS to produce more general classifiers because their condition part can cover large area of the problem space with more general hyper rectangle. So, NLCS can achieve better performance in classification problems with less number of evolved rules rather than UCS.

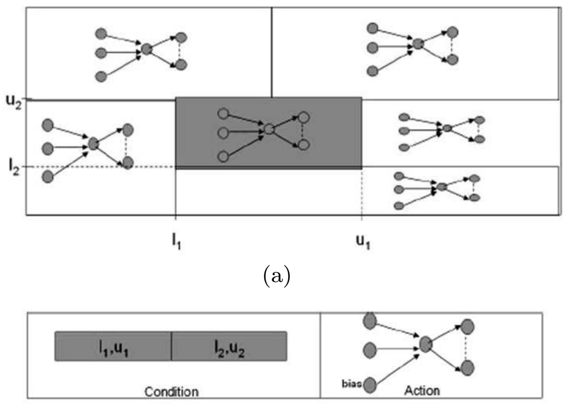


Fig. 10 An example of a classifier of NLCS in an arbitrary problem space directly drawn from (Dam et al., 2008). a) Visualization of the covered area by NLCS's classifiers. b) The genotype of the classifier highlighted in (a) which has a neural network in its action part.

3.9 Tile Coding Based Representation

Lanzi et al. had extended XCSF with tile coding prediction (Lanzi et al. 2006). Tile coding is one of the most common and successful methods to tackle complex environment in reinforcement learning realm (Sutton and Barto, 1998). In tile coding, the problem space is mapped into a set of overlapping tilings; each tiling partitions the input space into a set of nonoverlapping hyper-rectangles named tiles. Classifiers in XCSF with tile coding prediction have two additional parameters; the number of tilings and their resolution. The extended system can adapt these parameters through GA. Also the usual linear prediction function in XCSF is replaced with a tile coding approximator. So, the weight vector of each classifier contains the parameters related to each tile. Figure 11 shows how a classifier with two tilings partitions the input space and which tiles (states) could match the specified input. The new

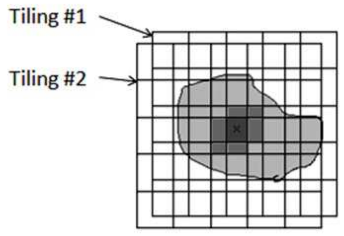


Fig. 11 Partitioning the problem space with a tile coding based classifier which suggests two way of tiling.

extended XCSF was tested on three multistep problems taken from the reinforcement learning literature: the 2D Gridworld (Boyan and Moore, 1995), the puddle world (Boyan and Moore, 1995), and the mountain car (Sutton, 1996). Indeed, such XCSF evolves an ensemble of tile coding approximators, each one on the problem subspace, instead of the typical monolithic approximator. The reported results showed that XCSF with tile coding can always reach an optimal solution and converge faster than XCSF with linear approximation.

4 Relative Strength and Weakness of Knowledge Representation Techniques

Each of knowledge representation techniques discussed in previous section its unique strengths but also weaknesses. For solving a given problem effectively by a rule based system such as LCS and XCS in particular, it is important to know which knowledge representation technique is the well suited for identifying the problem regularities. In other words, the first step in solving a problem is identifying the problem area to choose a proper representation which can cover whole problem space completely and embrace its complexity effectively. In this section, we present a few simple problem settings and analyze the relative strength and weakness of each discussed technique in handling such problems. Let us consider the following categories of problems. It is worth mentioning that we name each category based on the decision boundaries shapes and the complexity of the problem regularities.

Theoretical Problems: the main aim of this kind of problems is analyzing the behavior of the learning system in solving a problem without facing the complexity of environments. In other words, in these problems researchers try to choose a simple environment with specific properties. Hence, simple techniques for knowledge representation such as ternary for bit string inputs, and, interval based representation for real inputs could be a good choice.

Orthogonal Problems: in these problems the decision boundaries or regularities which are interesting to be modeled are parallel with the main axises. Figure 12 shows checkerboard problem, a sample problem of this problem category.

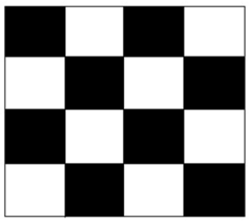


Fig. 12 Checkerboard problem: a classification problem data set. The positive instances belong to white region and the negative instances come from black region. This data set has axis-parallel boundaries.

Most of discussed knowledge representation techniques will handle this kind of problems. But interval based representation is well suited because it is a simple and popular technique and also it might be easier to apply compact rule set methods on the evolved rule set. In other words, since classifiers with interval based representation partition the problem space by defining axis-parallel hyper-rectangles of different sizes, this representation has an implicit ability to cover axis-parallel boundaries. But it is important to note that those problems with not only axis-parallel regularities but also continuous action can be handled using other representations such as neuro-fuzzy and neural network based representation. On the other way round, interval based representation will be able to handle such problems with continuous action if another independent method is used to produce a continuous action.

Oblique Problems: this category contains problems with oblique decision boundaries. Figure 13 shows several specimens of the recent type.

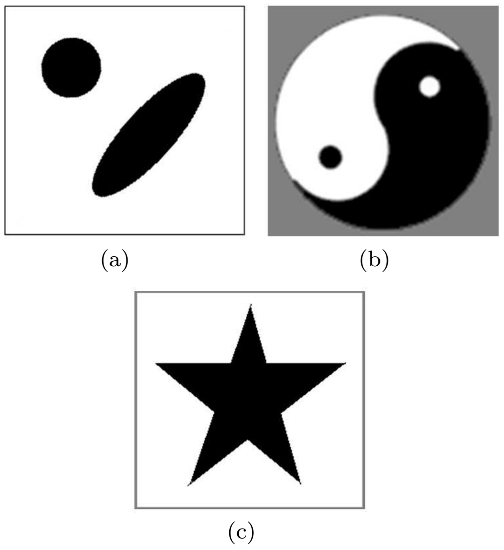


Fig. 13 Sample problems of Oblique Problems. Black region belongs to negative class and white region belongs to positive region. The name of these problems are; (a) Cycloid, (b) Tao (Llorá and Guiu, 2001) and (c) Pentagram (Ji and Dasgupta, 2004).

If we have enough information about the regularities of problems, it is better to use techniques which can intuitively fit the boundaries properly. To elucidate, consider Cycloid and Tao data sets which have oblique cycloidal boundaries. A proper choice for such boundaries might be ellipsoidal based representation due to its natural cycloidal shapes. On the contrary, convex hull based representation can model the cornered decision boundaries of Pentagram data set adequately due to providing asymmetric shape and having higher flexibility in covering complex areas.

Heterogeneous Problems: This category contains the problems in which have complex regularities or in other world have heterogeneous decision boundaries. To handle such problems successfully as if the decision boundaries can be modeled more precisely, obviously one has to choose a technique among more complex representation with ability to cover complex problem spaces. Since knowledge representation techniques such as GP-like and neural network based representation are general purpose and able to represent arbitrary regularities; they are also the methods of choice when the

Table 2 Existing approaches for knowledge representation in XCS with their main advantageous, disadvantageous, and the corresponding problem domains.

Rule representation (Systems)	**Advantageous**	**Disadvantageous**	**Problem Domain**
Interval Based Representation (XCSR, XCSI)	– Applicable to real valued environments. – Simple and easy to implement. – Easy to analyze. – Well-suited in problems which contain axis-parallel boundaries.	– Difficult to match oblique decision boundaries. – Limited to particular and symmetric shape.	– Real-Valued/ Integer-Valued problems (Wilson, 2000a) – Data Mining (Wilson, 2001b; Wilson, 2001; Bernadó-Mansilla et al., 2001; Gao et al., 2005; Gao et al., 2007; Gao et al., 2006) – Function Approximation (Wilson, 2002; Hamzeh and Rahmani, 2005; Hamzeh and Rahmani, 2007) – Robotics (Hurst and Bull, 2004) – Medical Data Mining-Ensemble Learning (Gao et al., 2007) – Classification - Large Data Sets (Llorá et al., 2007,Dam et al., 2005b) – Environment Navigation (Lanzi and Loiacono, 2007)
Ellipsoidal Based Representation	– Applicable to real valued environments. – Well-suited in problems wherein the dimensional dependencies are unknown.	– More parameters would be evolved in applying high dimensional problems. – Compaction method is needed to reduce the size of evolved rule set. – Limited by being a particular and symmetric, if orientable, shape.	– Function Approximation [Butz, 2005; Butz et al., 2006; Butz et al., 2008]
Convex Hull Based Representation	– Applicable to real valued environments. – Variable length representation. – Fine ability to identify more complex regions. – Generalization of interval and ellipsoidal representation. – Converge faster.	– Hard to find set of point to define a convex region. – Hard to define GA operators. – The number of points needed by convex hulls is exponential with dimensionality.	– Function Approximation[Lanzi and Wilson, 2006]
Fuzzy Logic Based Representation (Fuzzy LCS, ELF, LFCS, FIXCS, Fuzzy-XCS, Fuzzy-UCS)	– Applicable to real valued environments. – Capability to obtain maximal generalization(representation of the fuzzy rule set as compact as possible) – Fine interpretability of fuzzy rules. – Supporting real valued action. – Robustness to noisy input. – Handling mixed attribute and missing value.	– Producing only a limited number of possible rules. – Less freedom in domain knowledge representation and extraction. – Hard to make it flexible by using self-adapting mechanism.	– Classification [Valenzuela-Rendón, 1991; Orriols-Puig et al., 2007; Orriols-Puig et al., 2008a; Orriols-Puig et al., 2008b] – Robotics [Bonarini, 1993; Bonarini, 1994] – Epidemiologic Classification [Walter and Mohan, 2000] – Reinforcement Problems [Casillas et al., 2004; Casillas et al., 2005; Casillas et al., 2007; Bonarini, 1998] – Function Approximation [Casillas et al., 2004; Casillas et al., 2007]
First Order Logic Based Representation (FOXCS)	– Present the complex relationships among the attributes of a task domain	– Hard to define GA operators.	– Relational domains [Mellor, 2005] – ILP and RRL tasks [Mellor, 2006; Mellor, 2008] – Classification [Mellor, 2006; Mellor, 2008]

Messy Code Representation (XCSm)	– Variable length representation. – Well-suited in spars problems.	– Not suited for real valued problems.	– Multi step problems [Lanzi, 1999]
GP-like Representation (XCSL, XCSF-GEP, DGP-XCS)	– Applicable to real valued and nominal problems. – Able to represent arbitrary regularities. – Offer greater transparency. – Have the ability to ignore unneeded inputs or add ones that become relevant.	– Applicable to real valued and nominal problems. – Able to represent arbitrary regularities. – Offer greater transparency. – Have the ability to ignore unneeded inputs or add ones that become relevant.	– Function Approximation [Wilson,2008;] – Multi step problems [Preen and Bull, 2009; Lanzi and Perrucci, 1999; , Lanzi, 2003; Cielecki and Unold, 2007] – Data Mining [Lanzi, 2001a] – Learning Context-Free Language [Unold, 2005; Unold and Cielecki, 2005; Unold, 2007]
Neural Networks Based Representation (X-NCS, NCS, NLCS, XCSFNN, X-NFCS)	– Applicable to real valued environments. – Able to represent arbitrary regularities. – Able to apply on problem with continuous action space. – Able to produce more general classifier.	– Low interpretability – Every classifier must accept inputs from all variables, whereas this might not be necessary for some regularities.	– Function Approximation [Bull and O'Hara, 2002; Loiacono and Lanzi, 2006] – Multiple Domains [[Bull and O'Hara, 2002; O'Hara and Bull, 2005] – Robotics [Hurst and Bull, 2004] – Classification [Dam et al., 2008]
Tile Coding Based Representation (XCSF-RTC)	– Applicable to real valued environments. – XCSF with tile coding prediction converge faster than XCSF with linear approximation.	– Hard to evolved the number of tiling and their resolutions.	– Reinforcement Problems [Lanzi et al., 2006]

problem has not only complex regularities but also unknown dimensional independencies.

Problems with mixed attributes: Hitherto, real valued problems and bit string ones are discussed independently while mixed attributed problems are more common in real word applications. A straightforward solution is considering the problem as two subproblems: one is real valued problem with real valued attribute and the other is nominal problem only containing nominal attributes. These subproblems can be solved individually by two different representation techniques like interval based and messy code representation. Since not all mixed attributed problems can be divided into such subproblems, a representation which can handle both kind of attribute indiscriminately would be useful in such problems. GP-like and fuzzy based representations are some examples of those representation techniques.

In addition to what has already been mentioned, each representation technique has its own remarkable properties which provide an appropriate solution for particular problems. Table 2 summarized the existing knowledge representation approaches, some of their main advantageous, disadvantages, and the problem domain(s) for which the representation was designed and/or tested. We expect that this table to be a roadmap for choosing the best technique among wide varieties of existing knowledge representation approaches proposed so far.

5 Summary and Conclusion

In this study, we focus on the progress of XCS as a rule based learning system with a special consideration into the improvement of one of its most important components known as knowledge representation. As this component seeks prominent role in generalization capacity of the system and impacts on rule based system in terms of efficiency and efficacy, the past decade has seen a growing interest in proposing different knowledge representation techniques in LCS domain in general and XCS in specific. Here, after reviewing some basic information, we survey different knowledge representation techniques proposed in Michigan LCS realm and grouping them into different categories based on the classification approach in which they are incorporated. In each category, the underlying rule representation schema and the format of classifier condition to support the corresponding representation are presented. Furthermore, a precise explanation on the way that each technique partitions the problem space along with the extensive experimental results is provided. The main properties of knowledge representation technique, investigated in this paper, can be used as an illumination guideline to determine an appropriate knowledge representation technique for the domain in question. To shed light on these properties, a comparative analysis on some conventional problems is provided in general perspective.

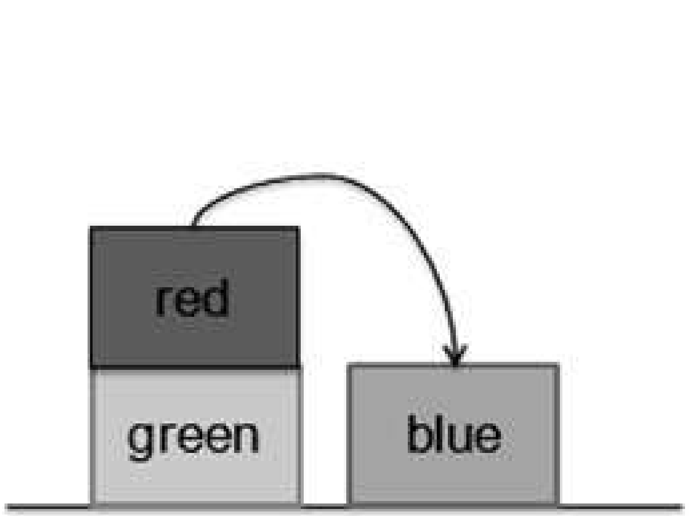
We hope that current review facilitates a better understanding of the different streams of research on this topic and underlies proper usage of LCS in many applications. In addition, since knowledge representation is a typical component of any rule based system like XCS, the current research can be a roadmap in other rule based systems to represent the problem regularities properly. Furthermore, we anticipate this survey to be interest to the LCS researchers and practitioners since it provides a guideline for choosing a proper knowledge representation technique for a given problem and also opens up new streams of research on this topic.

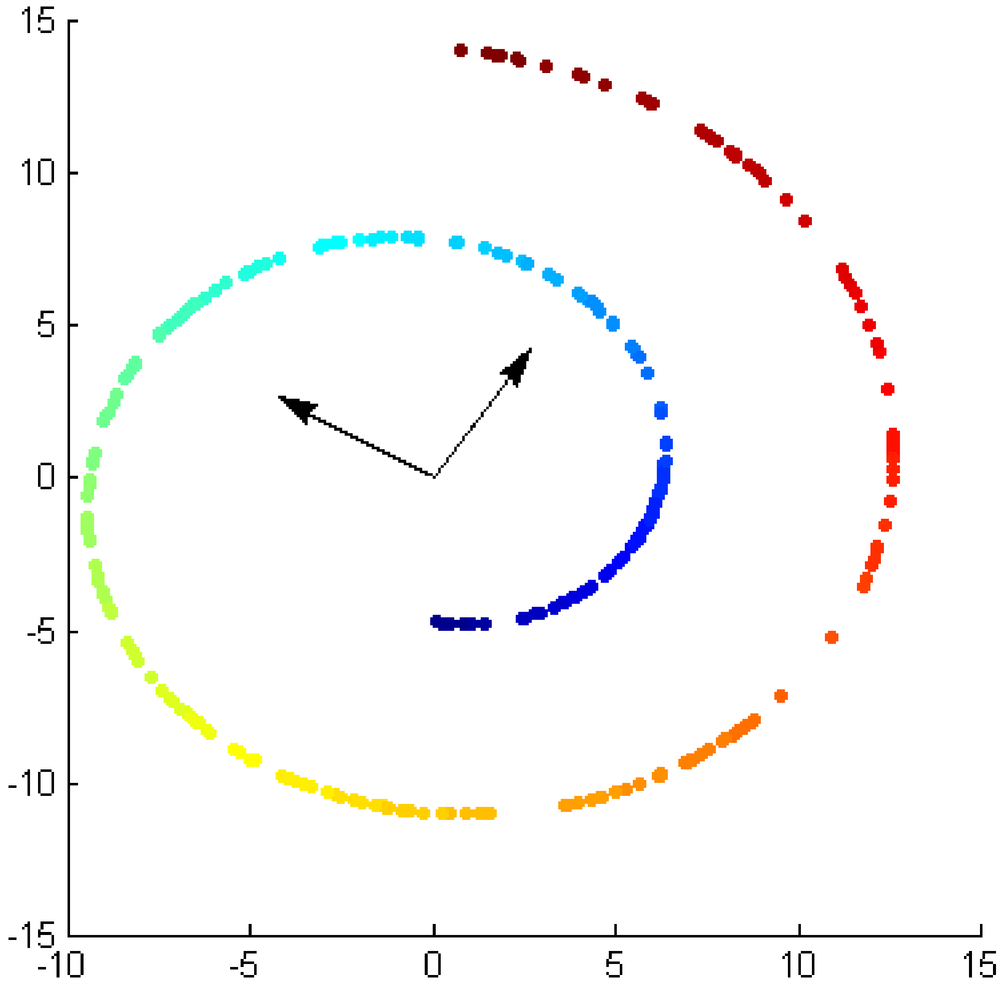
References

1. Bacardit J, Bernad*ó*-Mansilla E, Butz MV (2008) Learning Classifier Systems: Looking Back and Glimpsing Ahead. In Learn Clasif Sys From Found to Appl, pp 1-21.
2. Bernad*ó*-Mansilla E, Garrell J M (2003) Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks. Evol Comput 11(3): 209-238.
3. Bernad*ó*-Mansilla E, Llor*á* X, Garrell JM (2001) XCS and GALE: a comparative study of two learning classifier systems with six other learning algorithms on classification tasks. In Fourth Inter Workshop on Learn Clasif Sys IWLCS-2001,pp 337-341.
4. Bishop CM (1995) Neural Networks for Pattern Recognition. Oxford University Press, USA. Bonarini A (1993) ELF: learning incomplete fuzzy rule sets for an autonomous robot. Proc of EUFIT '93 ELITE Found, pp 69-75.
5. Bonarini A (1994) Some methodological issues about designing autonomous agents which learn their behaviors: the ELF experience. R. Trappl (ed.) Cybern and Sys Res 94:1435-1442.
6. Bonarini A (1996) Evolutionary learning of fuzzy rules: Competition and cooperation. Fuzzy Model: Paradig and Pract, pp 265-284.
7. Bonarini A (1998) Reinforcement distribution for fuzzy classifiers: A methodology to extend crisp algorithms. IEEE Intern Conf on Evol Comp, pp 265-284.
8. Bonarini A (2000) An introduction to learning fuzzy classifier systems. In Learn Clasif Sys From Found to Appl, pp 83-106.
9. Bonarini A, Bonacina C, Matteucci M (2000) Fuzzy and crisp representations of real valued input for learning classifier systems. In Learn Clasif Sys From Found to Appl, pp 107-124, Springer Verlag.
10. Bonarini A, Matteucci M (2007) FIXCS: a Fuzzy Implementation of XCS. FUZZ-IEEE 2007, pp 1-6.
11. Bonarini A, Matteucci M, Restelli M (2007) Learning Fuzzy Classifier Systems: Architecture and Exploration Issues. Intern J on Artif Intell Tools 16(2): 269-289.
12. Bonarini A, Trianni V (2001) Learning fuzzy classifier systems for multi-agent coordination. Info Sci 136(1-4): 215-239.
13. Boyan JA, Moore AW (1995) Generalization in reinforcement learning: Safely approximating the value function. Adv in Neural Info Proc Sys, pp 369-376.
14. Bull L (2004) Applications of learning classifier systems. Stud in fuzziness and soft comp. Springer.
15. Bull L, Bernad*ó*-Mansilla B, Holmes JH (2008) Learning classifier systems in data mining. Stud in Comp Intell, Springer.
16. Bull L, O'Hara T (2002) Accuracy-based neuro and neuro-fuzzy classifier systems. In GECCO '02: 905-911, San Francisco, CA, USA.
17. Bull L, Sha'Aban A, Tomlinson A, Addison J, Heydecker B (2004) Towards distributed adaptive control for road traffic junction signals using learning classifier systems. In: Appl of Learn Clasif Sys, pp 276-299.
18. Butz MV (2005) Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system. Proc of the 2005 conf on GECCO, pp 1835-1842.
19. Butz MV, Lanzi P, Wilson SW (2006) Hyper-ellipsoidal conditions in XCS: rotation, linear approximation, and solution structure. In Proc of the 8th GECCO, NY, USA, pp. 1457-1464.
20. Butz MV, Lanzi P, Wilson SW (2008) Function Approximation with XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction , IEEE Trans on Evol Comp 12(3): 355-376.
21. Butz MV, Wilson SW (2001) An Algorithmic Description of XCS, Adv in Learn Clasif Sys, pp. 253-272.
22. Casillas J, Carse V, Bull L (2004) Fuzzy xcs: An accuracy based fuzzy classifier system, in Proc of the XII Cong Esp sobre Tecnologiay Logica Fuzzy.
23. Casillas J, Carse V, Bull L (2005) Reinforcemen learning by an accuracy-based fuzzy classifier system with real-valued output, Proc I inter Workshop on Genet Fuzzy Sys, pp 44-50.
24. Casillas J, Carse B, Bull L (2007) Fuzzy-XCS: A Michigan Genetic Fuzzy System, IEEE Trans on Fuzzy Sys 15(4): 536-550.
25. Chomsky N (1956) Three models for the description of language, IRE Trans on Info Theory 2(2): 113-124.
26. Cielecki L, Unold O (2007) GCS with Real-Valued Input, Bio-insp Model of Cognetiv Task, pp 488-497.
27. Dam H, Abbass H, Lokan C (2005a) Be real! XCS with continuous-valued inputs, In Proc of the 2005 workshops on GECCO, NY, USA, pp 85-87.
28. Dam H, Abbass H, Lokan C, Yao X (2008) Neural-Based Learning Classifier Systems". Transactions on Knowledge and Data Engineering, vol. 20, no. 1: pp. 26-39.
29. Dam H, Shafi K, Abbass H (2005b) Can Evolutionary Computation Handle Large Dataset?, Tech Rep AI, pp 5-9.
30. De Jong KA (1988) Learning with genetic algorithms: An overview, Mach Learn, 3 :121-138.
31. Furuhashi T, Nakaoka K, Uchikawa Y (1994) Suppression of excess fuzziness using multiple fuzzy classifier systems, In Proc of the 3th IEEE Intern Conf on Fuzzy Sys, pp 411-414.
32. Goldberg DE (1989) Genetic Algorithms in Search, Optimization, and Machine Learning, Addision-Wesley Publishing Company, INC.
33. Goldberg DE, Korb B, Deb K (1989) Messy genetic algorithms. Motivation, analysis and first results. Complex Sys, 3: 493-530.

34. Gao Y, Huang J, Rong H, Gu D (2005) Learning classifier system ensemble for data mining. Proc of the 2005 workshops on GECCO, 63-66.
35. Gao Y, Huang J, Rong H, Gu D (2007) LCSE: Learning Classifier System Ensemble for Incremental Medical Instances, Learn Clasif Sys, pp 93-103.
36. Gao Y, Wu L, Huang J (2006) Ensemble Learning Classifier System and Compact Ruleset, Simu Evol and Learn, pp 42-49.
37. Hamzeh, A. and Rahmani A (2005) An Evolutionary Function Approximation Approach to Compute Prediction in XCSF. Mach learn : ECML, 3720:584-592.
38. Hamzeh A, Rahmani A (2007) A New Architecture of XCS to Approximate Real-Valued Functions Based on High Order Polynomials Using Variable-Length GA, In III Intern Conf on Nat Comp.
39. Hayes-Roth F (1985) Rule-based systems. Commun of the ACM 28: 921-932.
40. Holand JH (1976) Adaptation, Prog in Theor Bio 4: 263-293.
41. Holand JH (1986) "Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. Mach learn an artif intell approach 2: 593-623.
42. Holland JH, Reitman JS (1978) Cognitive systems based on adaptive algorithms Reprinted in: Evolutionary computation. The fossil record. IEEE Press, New York.
43. Howard DG, Bull L, Lanzi PL (2008) Self-adaptive constructivism in Neural XCS and XCSF. Proc of GECCO 2008, Atlanta, USA, pp 1389-1396.
44. Hurst J, Bull L (2004) A self-adaptive neural learning classifier system with constructivism for mobile robot control. Parallel Problem Solving from Nature - PPSN VIII, LNCS 3242: 942-951.
45. Ishibuchi H, Nakashima T, Murata T (1999) "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE Trans on Sys, Man, and Cybern-Part B: Cybernetics, 29(5): 601-618.
46. Jang JS, Sun CT (1995) Neuro-fuzzy Modeling and Control. Proc of the IEEE 83(3): 378-406.
47. Ji Z, and Dasgupta D (2004) Real-valued negative selection algorithm with variable-sized detectors. GECCO 2004. LNCS 3102: 287-298.
48. Koza J R (1992) Genetic Programming: on the Programming of Computers by Means of Natural Selection. MIT Press.
49. Langton WB, Soule T, Poli R, Foster JA (1996) The evolution of size and the shape. Adv in Genet Program 3: 163-190. MIT Press.
50. Lanzi PL (1998) Adding Memory to XCS. In Proc of the IEEE Conf on Evol Comp, IEEE Press, pp 609-614.
51. Lanzi PL (1999) Extending the representation of classifier conditions part I: from binary to messy coding. Proc of GECCO 1999, pp 337-344.
52. Lanzi PL (2001a) Mining interesting knowledge from data with the XCS classifier system. Proc of GECCO 2001, pp 958-965.
53. Lanzi PL (2001b) Generalization in the XCS Classifier Systems with Symbolic Representation. Technical Report 01, Dipartimento di Elettronicae Informazione.
54. Lanzi PL (2003) XCS with stack-based genetic programming. In CEC2003 , pp 1186-1191, IEEE Press.
55. Lanzi PL (2008) Learning classifier systems: then and now. Evol Intell 1: 63-82.
56. Lanzi PL, Loiacono D, Wilson SW, Goldberg DE (2006) Classifier prediction based on tile coding. GECCO 2006, ACM Press, 2: 1497-1504.
57. Lanzi P L, Loiacono D (2007) Classifier systems that compute action mappings. In Proc of GECCO 2007 , ACM Press, pp 1822-1829.
58. Lanzi PL, Perrucci A (1999) Extending the representation of classifier conditions part II: From messy coding to S-expressions. Proc of GECCO 1999 1: 345-352.
59. Lanzi PL, Stolzmann W, Wilson SW (eds) (2000) Learning classifier systems: from foundations to applications. LNCS 1813, Springer.
60. Lanzi P L, Wilson SW (2006) Using convex hulls to represent classifier conditions. In Proc of GECCO 2006, ACM Press, pp 1481-1488.
61. Llorá X, Guiu JMG (2001) Inducing partially-defined instances with evolutionary algorithms. ICML, pp 337-344.
62. Llorá X, Reddy R, Matesic B, Bhargava R (2007) Towards better than human capability in diagnosing prostate cancer using infrared spectroscopic imaging. In Proc of GECCO 2007, ACM Press, pp 2098-2105.
63. Loiacono D, Lanzi P (2006) Evolving Neural Networks for Classifier Prediction with XCSF. In ECAI Neural Networks for Classifier Prediction with XCS, pp 36-40.
64. Marin-Blázquez JG, Martnez-Prez G (2008) Intrusion detection using a linguistic hedged fuzzy-XCS classifier system. Soft Comp 13(3): 273-290.
65. Marin-Blázquez JG, Martinez Perez G, Gil Perez M (2007) A Linguistic Fuzzy-XCS classifier system. In Fuzzy Sys Conf, IEEE Int, pp 1-6.
66. Marin-Blázquez JG, Shen Q (2008) A fuzzy-XCS classifier system with linguistic hedges. Comp Intell Res 4(4): 329-341.
67. Mellor D (2005) A first order logic classifier system. Proc of GECCO 2005, ACM Press, pp 1819-1826.
68. Mellor D (2006) Extending XCS with Representation in First Order Logic. Proc IWLCS 2006, Seattle, MA, USA.
69. Mellor D (2008) A Learning Classifier System Approach to Relational Reinforcement Learning. Learn Clasif Sys, pp 169-188.
70. Moscato P (1989) On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Caltech Concur Comp Program ,report 826.
71. Nakaoka K, Furuhashi T, Uchikawa Y (1994) A study on apportionment of credits of fuzzy classifier system for knowledge acquisition in large scale systems. In Proc of 3th IEEE Int Conf on Fuzzy Syst, pp 1797-1800.
72. Nomura Y, Ikebukuro K, Yokoyama K, Takeuchi T, Arikawa Y, Ohno S, Karube I, Valenzuela-Rendón M (1998) Reinforcement learning in the fuzzy classifier system. Expert Sys with Appl 14: 237-247.
73. O'Hara T, Bull L (2005) A Memetic Accuracy-based Neural Learning Classifier System. In Proc of the IEEE Congr on Evol Comp, IEEE Press, pp 2040-2045.
74. Orriols-Puig A, Casillas J, Bernadó-Mansilla E (2007) Fuzzy-UCS: preliminary results. Proc of GECCO 2007, ACM Press, pp 2871-2874.
75. Orriols-Puig A, Casillas J, Bernadó-Mansilla E (2008a) Fuzzy-UCS: a Michigan-style learning fuzzy-classifier system for supervised learning. IEEE Trans on Evol Comp 13(2): 260-283.
76. Orriols-Puig A, Casillas J, Bernadó-Mansilla E (2008b) Approximate versus linguistic representation in fuzzy-UCS. Proc of the 2008 Int Workshop on Hybrid Artif Intell Sys HAIS'08, LNAI 5271: 722-729.
77. Parodi A, Bonelli P (1993) A new approach to fuzzy classifier systems. Proc of the 5th Int Conf on Genet Algorithms, pp 223-230.
78. Patel MJ, Dorigo M (1994) Adaptive Learning of a Robot Arm. Evol Comp, AISB workshop selected papers, LNCS 865, Springer, pp 180-194.

79. Perkis T (1994) Stack-based genetic programming. Proc of the 1994 IEEE World Congr on Comp Intell 1: 148-153.
80. Preen R, Bull L (2009) Discrete dynamical genetic programming in XCS. Proc of GECCO 2009. ACM Press, pp 1299-1306.
81. Schulenburg S, Ross P (1999) An Adaptive Agent Based Economic Model. Learn Clasif Sys, pp 263-282.
82. Schuurmans D, Schaeffer J (1989) Representational difficulties with classifier systems. Proc of the 3rd Int Conf on Genet Algorithms (ICGA89), pp 328-333.
83. Sigaud O, Wilson SW (2007) Learning Classifier Systems: A Survey. Soft Comp - A Fusion of Foundations, Methodologies and Applications 11(11): 1065-1078.
84. Smith SF (1980) A learning system based on genetic adaptive algorithms. Doctoral thesis, Pittsburgh, PA, USA.
85. Smith SF (1983) Flexible learning of problem solving heuristics through adaptive search. 8th Int Joint Conf on Artif Intell, Los Altos, pp 421-425.
86. Soule T, Foster JA, Dickinson J (1996) Code growth in genetic programming. Genet Program Proc First Annual Conf , MIT Press, pp 215-223.
87. Stone C, Bull L (2003) For real! XCS with continuous-valued inputs. Evol Comp, 11(3): 299-336.
88. Studley M (2005), Learning classifier systems for multi-objective robot control. Ph.D. thesis, Faculty of Computing, Engineering and Mathematics University of the West of England. Learning Classifier Systems Group Technical Report UWELCSG06-005.
89. Studley M, Bull L (2005) X-tcs: accuracy-based learning classifier system robotics. Proc of IEEE Congr on Evol Comp, CEC 2005, pp. 2099-2106.
90. Sutton RS (1996) Generalization in reinforcement learning: Successful examples using sparse coarse coding. Neural Info Processing Sys, MIT Press, pp 1038-1044.
91. Sutton RS, Barto AG (1998) Reinforcement learning-an introduction. MIT Press, Cambridge. Unold O (2005) Context-free grammar induction with grammar-based classifier system. Arch of Rchives of Control Sci 15(4): 681.
92. Unold O (2007) Grammar-Based Classifier System for Recognition of Promoter Regions. Adapt and Nat Comput Algorithms, LNCS 4431:798-805.
93. Unold O, Cielecki L (2005) Grammar-based Classifier System. Issues in Intell Sys: Paradigms, EXIT, Warsaw, pp 273-286. Urbanowicz RJ, Moore JH (2009) Learning Classifier Systems: A complete introduction, Review, and Roadmap, J of Artif Evol and Appl, pp 1-25.
94. Valenzuela-Rendón M (1991) The fuzzy classifier system: A classifier system for continuously varying variables. In 4th ICGA, pp 346-353.
95. Vargas P, Filho C, Zuben FV (2004) Application of learning classifier systems to the on-line reconfiguration of electric power distribution networks. Appl of Learn Classif Systems. Studies in fuzziness and soft computing, Springer, Heidelberg, pp. 260-275.
96. Velasco J (1998) Genetic-based on-line learning for fuzzy process control. Int J of Intell Sys, 13: 891-903. Walter D, Mohan C (2000) ClaDia: a fuzzy classifier system for disease diagnosis. Proc of the 2000 cong on Evol Comp 2:1429-1435.
97. Wilson SW (1994) ZCS: a zeroth level classifier system. Evol Comp 2(1): 1-18.
98. Wilson SW (1995) Classifier Fitness Based on Accuracy. Evol Comp 3(2): 149-175.
99. Wilson SW (2000a) Get real! XCS with continuous-valued inputs. Learn Clasif Sys, From Found to Appl, LNAI-1813, pp 209-219.
100. Wilson SW (2000b) Mining oblique data with XCS. Proc of the Third Int Workshop (IWLCS-2000), LNAI, pp 158-174.
101. Wilson SW (2001) Compact Rulesets from XCSI. Revised Papers from the 4th Int Workshop on Adv in Learn Clasif Sys, pp 197-210.
102. Wilson SW (2002) Classifiers that approximate functions. Natural Comp 1(2-3): 211-234.
103. Wilson SW (2008) Classifier Conditions Using Gene Expression Programming. IWLCS 2007, LNAI 4998, pp 206-217.
104. Wong SY, Schulenburg S (2007) Portfolio allocation using XCS experts in technical analysis, market conditions and options market. GECCO 2007, pp 2965-2972.
105. Wyatt D (2004) Applying the XCS Learning Classifier System to Continuous-Valued Data-mining Problems. Technical Report UWELCSG05-001, Learning Classifier Systems Group, University of the West of England, Bristol, UK.
106. Zadeh L A (1965) Fuzzy sets. Info and Control 8:338-353.
107. Zadeh L A (1973) Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans on Sys, Man, and Cybern 3: 28-44.


red
green
blue


15
10
5
0
-5
-10
-15
-10
-5
0
5
10
15